# Lab Task 3: Singly Linked List

## Code Explanation:

This program implements a Singly Linked List with two types of insertions:  
- \*\*Insert at Start\*\*: A new node is created and added at the beginning of the list.  
- \*\*Insert at End\*\*: A new node is created and added at the end of the list.  
- The `display()` function prints the linked list after each insertion.  
- The `main()` function inserts values `{3, 1, 5, 7}` in a sequence and prints the list.

## Code:

#include <iostream>  
using namespace std;  
  
class Node {  
public:  
 int data;  
 Node\* next;  
 Node(int val) : data(val), next(nullptr) {}  
};  
  
class SinglyLinkedList {  
public:  
 Node\* head;  
 SinglyLinkedList() : head(nullptr) {}  
  
 void insertAtStart(int val) {  
 Node\* newNode = new Node(val);  
 newNode->next = head;  
 head = newNode;  
 display();  
 }  
  
 void insertAtEnd(int val) {  
 Node\* newNode = new Node(val);  
 if (!head) {  
 head = newNode;  
 } else {  
 Node\* temp = head;  
 while (temp->next) {  
 temp = temp->next;  
 }  
 temp->next = newNode;  
 }  
 display();  
 }  
  
 void display() {  
 Node\* temp = head;  
 while (temp) {  
 cout << temp->data << " -> ";  
 temp = temp->next;  
 }  
 cout << "NULL" << endl;  
 }  
};  
  
int main() {  
 SinglyLinkedList list;  
 int values[] = {3, 1, 5, 7};  
   
 list.insertAtStart(values[0]);  
 list.insertAtStart(values[1]);  
 list.insertAtEnd(values[2]);  
 list.insertAtEnd(values[3]);  
   
 return 0;  
}

## Output:

```  
3 -> NULL  
1 -> 3 -> NULL  
1 -> 3 -> 5 -> NULL  
1 -> 3 -> 5 -> 7 -> NULL  
```

## Time Complexity:

- \*\*Insert at Start:\*\* O(1) (Constant time insertion at the beginning)  
- \*\*Insert at End:\*\* O(n) (Traversal required to reach the end of the list)